User's Manual For:

DOSPLUS III Ver 3.3

Disk Operating System for the TRS-80 Model III

IMPORTANT NOTICE:

Model III TRSDOS disks and Model III DOSPLUS disks are NOT diskette read/write compatible. Refer to the section of the manual labeled **CONVERT** before losing one's cool and performing atrocities upon your system's diskette.

TABLE OF CONTENTS

Section	Item	Page #
Introduction		1
	BACKUP	2
	FILESPECS	3–4
Library Commands		4
	APPEND	4–5
	ATTRIB	5
	AUTO	6
	BUILD	6
	CLEAR	6
	CLOCK	7
	CONFIG	7–8
	COPY	8
	CREATE	8–9
	DATE	9
	DEBUG	9–10
	DEVICE	10
	DIR	10–11
	DO	12
	DUMP	12
	FORCE	12
	FORMS	12–13
	FREE	13
	KILL	14
	LIB	14
	LIST	14
	LOAD	14
	PAUSE	14
	PROT	14–15
	RENAME	15
	RS232	15
	TIME	15
	VERIFY	15
Utilities		16
	BACKUP	16
	CLRFILE	16
	COPY1	17
	CRUNCH	17
	DISKDUMP	17–18
	DISKZAP	18
	Introduction	18
	MODE	19
	ZERO	19
	COPY	19
	PRINT	20
	VERIFY	20
	FORMAT	21
	DISPLAY	21

TABLE OF CONTENTS (Cont.)

Section	Item	Page #
Utilities (Cont.)		
	FORMAT	21–22
	PURGE	22
	RESTORE	22
	SPOOL	23
	TRANSFER	23
	MAP	23–24
BASIC		24
	Introduction	24
	Entering BASIC	24–25
	BASIC Commands	25
	CMD	25
	DI	25
	DU	25
	EDIT	25–26
	OPEN"E"	26
	OPEN"R"	26
	RENUMBER	26
	TAB	27
	TRACE	27
	REFERENCER	27
	CMD"M"	27
	,V (Variable Save)	27
	Search & Replace	27–28
	TBASIC	28
	Compatibility	28
CONVERT		28–29
Patch Program Instruction Sheet		29–30
Sample Programs		31–33
	Variable Length Records	31–32
	Directory Subroutine	33
	Date Input Subroutine	33

DOSPLUS Ver. 3.3 USER'S MANUAL

INTRODUCTION:

The purpose of this manual is to provide you, the user, with all the information necessary to use **DOSPLUS** and all the utilities that come with it. This is a User's Manual, not a Technical Manual.

DOSPLUS is the culmination of several years of frustration with the other operating systems available for the **TRS-80***, and a desire to write a truly superior operating system for this microcomputer. **DOSPLUS** is the most powerful Disk Operating System **(DOS)** on the market today and the easiest to use. Several important things need to be stressed at this time:

1. **DOSPLUS** is designed to be "error free." It will not "hang-up" or aimlessly re-boot your system. **DOSPLUS** is also fast! We encourage you to compare its speed with that of its competitors. **DOSPLUS** is significantly faster than any other **DOS** currently available.

2. It is not necessary to transfer drivers or routines into high memory because the routines and drivers are already down low in memory, as part of the operating system, where they belong. However, one buffer in high memory is allocated during a **DO** (See **Library Commands**) for Disk I/O. **BASIC's** new **CMD" "** feature will work with any **DOS** command and control is returned to **BASIC** at the completion of the command. This means you can use **DOS** commands as **BASIC** program steps and return to **BASIC** at the next statement without loss of your data. There is no need for **POKE**ing machine language routines into high memory because you can write a machine language program, store it on disk, execute it from **BASIC** (via **CMD" "**) and return to **BASIC** with your program intact. The ramifications of this feature are endless. **TBASIC** loads in the usual memory locations which will allow you to run those long **BASIC** programs. However, **TBASIC** does not afford you the convenience of the **CMD" "** feature. These two **BASICS** are compatible except for the **CMD" "** feature.

3. **DOSPLUS** has several powerful built-in features. To enter a lower case letter, hold the **«SHIFT»** key down and press the desired letter keys, or enter the typewriter mode by pressing **«SHIFT»«0»**, and automatically type lower case letters. You then use the **«SHIFT»** key for upper case letters as on a standard typewriter. It also combines a powerful debounce routine with a repeating keyboard feature. To use the repeat key function, simply hold down the desired key and it will automatically repeat. However, you only get the repeating keys feature when you want them. **DOSPLUS** also has an easy-to-use screen printer routine. Simply press **«SHIFT» «CLEAR»** and the information on the screen will be printed on your printer. Unlike other screen printer routines, if you engage the **DOSPLUS** screen printer routine and the printer is not ready, it will not "hang-up" your computer. It simply ignores you. This avoids inadvertent "hang-ups" that plague other screen print routines. To abort a screen print, hold down the **«BREAK»** key.

4. Many other Disk Operating Systems advertise variable length records. Most of them don't work, and if they do, they have bugs that prevent them from working properly. We have banished these frustrations forever. Our variable length records work first time, every time. To use the variable length records, all you need do is specify the number of bytes in a record when you open a file in **BASIC** (i.e., **OPEN "R",1,"FILESPEC",64**). Also, all of our library commands perform their advertised functions. If you follow the instructions in this manual, you should never have a problem.

NOTE: In this manual all **DOSPLUS** prompts are printed in **BOLDFACE;** your responses are in normal weight type.

* **TRS-80** is a trademark of Tandy Corp.

MAKING A DOSPLUS BACKUP:

Before you do anything else, you should make a working copy of the **DOSPLUS** disk. In this way, should anything happen to your working disk, you can always make a new copy. After you have made a working copy, file the original **DOSPLUS** disk in a safe place away from magnetic fields.

To make a backup of your **DOSPLUS** disk, use the following procedure.

1. Power up your computer in the usual manner. (Remember, NEVER power up your computer with a disk in the drive.)
2. Place the **DOSPLUS** disk into drive 0 and press the **«RESET»** button on the right front of the keyboard. The display will read:

DOSPLUS — MODEL III OPERATING SYSTEM — VER. 3.3 XXK
COPYRIGHT ©1981, MICRO-SYSTEMS SOFTWARE INC.

DOS PLUS
#

NOTE: The **XXK** will be the memory size of your computer. If you have a 32K machine, it will read **32K.**

If you have more than one drive, place a blank disk into drive 1. If you have only a single disk drive, have a blank disk ready and **DOSPLUS** will tell you when to insert it into drive 0.

Type **BACKUP** and press **«ENTER».**

The screen will clear and will then display:

DOSPLUS — Model III Backup Utility — Ver. 3.3
Copyright ©1981, Micro-Systems Software Inc.

If you have only 1 drive, type:

SOURCE DRIVE NUMBER ? **0 «ENTER»**
DESTINATION DRIVE NUMBER ? **0 «ENTER»**

If you have more than one disk drive, type:

SOURCE DRIVE NUMBER ? **0 «ENTER»**
DESTINATION DRIVE NUMBER ? **1 «ENTER»**

Now, type the date in **MONTH/DAY/YEAR (MM/DD/YY)** format. For example, for October 29, 1980, type:

BACKUP DATE (MM/DD/YY) ? **10/29/80 «ENTER»**

DOSPLUS will then make the **BACKUP.** First, it will format the blank disk. Then it will duplicate the contents of the **DOSPLUS** disk onto the disk it just formatted. If you are using only one drive, the **BACKUP** program will prompt you when to remove the **SOURCE** disk and insert the **DESTINATION** disk. You will also be asked to insert a **SYSTEM** disk, which in this case will also be the **SOURCE** disk. You may have to exchange disks several times to make a backup. When the process is completed, the message,

INSERT SYSTEM DISK «ENTER»

will flash on the screen. Simply press **«ENTER»** in response to this prompt and the **DOSPLUS** prompt will be displayed.

NOTE: The separator **TO** in **DOSPLUS** is optional. In the above example, you could have used the format:

APPEND DATA1:1 DATA:0 «ENTER»

WARNING: If you **APPEND** a **BASIC** program to the end of another when you load the combined program into the computer the line numbers will be merged.

ATTRIB — ATTRIB is used to change the attributes of a disk file. You can alter the following attributes:

1. Make a file visible/invisible.
2. Change access or update passwords.
3. Change the level of protection.

The format is:

ATTRIB FILESPEC:D (PARAMETER1,PARAM2,PARAM3) **«ENTER»**

Where the parameters can be:

I — Make a file visible/invisible.
A — Alter access password.
U — Alter update password.
P — Set protection level.

The protection levels are:

Level	Parameter	Function
0	**FULL**	No protection
1	**KILL**	Able to delete file
2	**RENA**	Rename, write, read, execute
3	**————**	Not used at this time
4	**WRIT**	Write, read, execute
5	**READ**	Read, execute
6	**EXEC**	Execute only
7	**NONE**	No access, can't be set by user

Access level 7 is set by Micro-Systems. It is for system files only. You cannot make use of it. The only function that can make use of these files is the **PURGE** command.

EXAMPLE 1 **ATTRIB TEST1/BAS (I) «ENTER»**

This will make the program **TEST1/BAS** invisible. However, if the program was already invisible, it will make it visible again.

EXAMPLE 2 **ATTRIB TEST1/BAS (A=**PASSWORD**) «ENTER»**

This will assign the specified password as the access password for the program **TEST1/BAS.**

EXAMPLE 3 **ATTRIB TEST1/BAS (U=**PASSWORD**) «ENTER»**

This will assign the specified password as the update password for the program **TEST1/BAS.**

NOTE: To remove a password, you need only to enter the parameter **A=** or **U=.**

EXAMPLE 4 **ATTRIB TEST1/BAS (U=PASSWORD,P=EXEC)**

This will set the protection level so that the program can be executed only. Remember, you must enter the *word* for the level of protection you desire and not the *digit.*

You can use more than one parameter at the same time. For example:

ATTRIB TEST1/BAS (I,A=PASSWORD,U=UPDATE, P=WRIT) «ENTER»

AUTO — This command will allow you to execute a system command or a program from system power up or re-boot. The format is,

AUTO COMMAND LINE

Where the command line is a library command or a machine language program name.

EXAMPLE **AUTO DIR**

This will automatically display the directory of the disk in drive 0 on power up. To deactivate this function, simply type **AUTO** without a parameter and it will be turned off.

BREAK — This feature allows you to disable/enable the **«BREAK»** key. The format is:

BREAK (OFF) to disable; and
BREAK (ON) or **BREAK** to re-enable

It may be used from within a program in Extended Disk **BASIC** via the **CMD** function. It will effectively prevent unwanted program interruption.

BUILD — This feature allows you to create a **DO** file. This file can contain a series of command lines up to 63 characters in length. These command lines can contain either **DOS** commands or **BASIC** program lines. This enables you to create and execute a simple program right from power up. The primary function of this feature is to allow you to enter a **BASIC** program right from power up. You can set the number of files, memory size, and execute the program without operator input. Another way to use this feature is to load the drivers for a serial printer, then load **BASIC**, and protect the memory location of the serial driver. The format is:

BUILD FILESPEC

If you don't specify a file extension, **DOSPLUS** will automatically add **/BLD** onto your filespec. You will then see:

TYPE IN UP TO 63 CHARS

At this point, type in a single command line (you can't have two commands on one line) and then press **«ENTER»**. **DOSPLUS** will then prompt you for the next command line. When you have finished entering commands, press the **«BREAK»** key in response to the prompt and your file will automatically be saved on the disk in drive 0.

EXAMPLE **BUILD START «ENTER»**
TYPE IN UP TO 63 CHARS
FORMS (S) «ENTER»
BASIC LEDGR/BAS.PASSWORD-F:3-M:61285 «ENTER»
TYPE IN UP TO 63 CHARS
«BREAK»

This will create a file on your disk named **START/BLD** and return you to **DOS.** To use this file type:

AUTO DO START

Now whenever you boot your system with this disk in drive 0, you will automatically load the Serial Printer Drivers, protect memory above 61285, open 3 files, and then run the **BASIC** program **LEDGR/BAS** that is password protected.

CLEAR — This command clears all user memory above **X'5700'** hex and then clears the screen. It enables you to have a clean slate of memory. **CLEAR** does a quick memory check. To execute this feature, type **CLEAR** and press **«ENTER»**.

NOTE: **CLEAR** does not affect the buffer allocated in high memory for the **DO** command and can be executed as part of a **DO** file.

CLOCK — This command will display the time in the upper right hand corner of the screen. The format is:

CLOCK (PARAMETER)

The only parameters for this command are **ON** and **OFF**. However, if a parameter is not typed, **DOSPLUS** will assume the parameter **ON**.

EXAMPLE **CLOCK (OFF) «ENTER»**

NOTE: The clock may lose some time when the disk is accessed.

CONFIG — This command allows you to custom-configure your system to your specific needs. Using this command, you can optimize your disk access procedures. You can modify the following items:

1. Number of tracks on a disk.
2. Track-to-track step rate.
3. High speed initialization.
4. Double-sided drive operation.
5. Printer Graphics Activation

The format is:

CONFIG (PARAMETER1,PARAMETER2,PARAMETER3,ETC) **«ENTER»**

Where you can have the following parameters:

TRKS — Tells your system the number of tracks that are on the system disk. This will cause the system to make backups using the specified number of tracks. If you specify 80 tracks it will even make 80 track backups of a 40 track disk.

STEP — Sets the head step rate. The faster the head moves the faster your disk I/O. The rate must be compatible with your disk drives. Some drives allow much faster access than others. The allowable step rates are 6, 12, 20, 30, and 40 milliseconds.

SPEED — If your computer has a high speed clock modification installed, **DOSPLUS** will operate with it (up to 4.0 MHz). Speed will output a byte to port 254 on power up. The value you set speed equal to (0–255) determines what will be output. If your speed-up mode does not use port 254 to initialize, speed will do you no good, and should be left at zero.

SIDES — This allows you to configure a drive as double- or single-sided. Using a double-headed drive may require a special drive cable. Consult the manual that came with your drive.

GPHON or **GPHOFF** — If your printer can print graphics this will turn it on or off. (No drive specified.)

NOTE: When you configure your system for double-headed drives you may use the side select option. Consult the section **NOTES ON FILESPECS** at the beginning of the manual for details.

If you type **CONFIG** without any parameters you will get a display of the current system configuration.

EXAMPLE #1 **CONFIG «ENTER»**

```
TRACKS = 40,SPEED = 00,GPHOFF
:0 STEP = 40MS, SIDES = 1
:1 STEP = 40MS, SIDES = 1
:2 STEP = 40MS, SIDES = 1
:3 STEP = 40MS, SIDES = 1
```

This is the way you received the original diskette.

EXAMPLE #2 **CONFIG :0 (TRKS = 80,STEP = 20) «ENTER»**

```
TRACKS = 80,SPEED = 00,GPHOFF
:0 STEP = 20MS, SIDES = 1
:1 STEP = 40MS, SIDES = 1
:2 STEP = 40MS, SIDES = 1
:3 STEP = 40MS, SIDES = 1
```

This will give you an 80 track system diskette stepping at 20 milliseconds only after you re-boot the system and then make a **BACKUP** of the diskette. Had you not changed tracks no **BACKUP** would have been necessary just to change **STEP**. Remember that this command makes the **GAT** think it has forty extra tracks. They aren't really there. So when you do a **FREE**, it will show the last forty tracks locked out. It will *not* allow you to write to these tracks! When you make a backup, **BACKUP** will read the **GAT** and format the destination disk to 80 tracks giving you a valid 80 track disk.

EXAMPLE #3 **CONFIG :2 (SIDES = 2,STEP = 20) «ENTER»**

```
TRACKS = 80,SPEED = 00,GPHOFF
:0 STEP = 20MS, SIDES = 1
:1 STEP = 40MS, SIDES = 1
:2 STEP = 20MS, SIDES = 2
:3 STEP = 40MS, SIDES = 1
```

With this example you just set drive #2 as a double headed drive and the **STEP** rate is now 20 milliseconds. **DIR :2B** will now give you the directory on the second side of drive #2. Remember, you must first have the hardware necessary. You will have up to eight directories, and files may not span sides.

EXAMPLE #4 **CONFIG (GPHON) «ENTER»**

This just told your computer not to change the graphics characters before sending them to the printer. **CONFIG (GPHOFF)** switches your computer back to normal. Normal is when the machine sends a "*" to the printer instead of a graphics character.

NOTE: For any drive other than zero, you can set the step rate and side configuration only. After you have used the **CONFIG** command to change the system configuration you *must* re-boot the system to power up in the new configuration.

COPY — This feature allows you to create a duplicate file. However, it will only work with multiple drives. It cannot be used by single drive users. Single drive users must use the **COPY1** utility program (see **COPY1**). The format is:

COPY FILESPEC :D **TO** FILESPEC :D

The first filespec and disk drive specifications are mandatory. However, the **TO** and second filespec are optional and may be omitted. For example,

COPY FILESPEC :D :D

Where the first (D) is the source drive and the second (D) is the destination drive.

EXAMPLE **COPY TEST1/BAS:0 TO TEST1/BAS:1 «ENTER»**

or

COPY TEST1/BAS:0 :1 «ENTER»

These will both create a file name **TEST1/BAS** on drive 1.

CREATE — This feature is used to create a file and preallocate disk space. The advantage of using **CREATE** is that it will speed up file access time during a write. This is possible because **DOSPLUS** does not have to periodically allocate space on the disk as it writes. Files that use preallocated disk space are generally less segmented on the disk and the disk drive head does not have to travel as much during disk I/O. The format is:

CREATE FILESPEC :D **(NRECS=N1,LRL=N2)**

NRECS is the number of records to allocate and **LRL** is the length of each of these records. **LRL** can be any length befween 1 and 255. 256 is fhe default value and may not be specified from the command line.

EXAMPLE **CREATE ACCNTS/DAT:1 (NRECS=1000,LRL=128) «ENTER»**

NOTE: If the filespec already exists, **CREATE** will abort and return to **DOSPLUS. CREATE** is not mandatory. It you don't preallocate disk space, **DOSPLUS** will do it automatically as more space is needed.

DATE — This teature allows you to display and set the current date. The format is:

DATE «ENTER»

This will display the current date and it requires no parameter.

DATE MM/DD/YY **«ENTER»**

This will set the date to a specified date where MM is a two-digit month, DD is a two-digit day and YY is a two-digit year. You must add a zero to all single-digit numbers.

EXAMPLE **DATE** 11/05/80 **«ENTER»**

This will sef the date fo November 5, 1980.

Once the date is set, **DOSPLUS** will use fhat dafe anytime it would normally request you to enter a date from the keyboard. For example, when making a backup copy of a disk, if the date is set, the date question will be skipped. Once the date is set, it will remain set until you re-boot the system by pressing **«RESET»** or by turning the compufer otf.

DEBUG — DEBUG is a powerful disk base monitor. With it you can examine any memory locafion. In both **RAM** and **ROM,** or any **CPU** register. You may also change the content of a **RAM** location or registfer. **DEBUG** is so powerful fhat if should be used with caution, because if is easy to accidentally destroy a program.

Unlike the other **DOSPLUS** commands, when you enable **DEBUG** you will nof see any noticeable change on the screen. This is because **DEBUG** is transparent to the execution ot your program and is only entered when called. There are three ways to call **DEBUG** when it has been enabled. They are:

1. Pressing **«SHIFT»«BREAK»** at any fime
2. Automatically atter a program has been loaded and before the first instruction has been executed.
3. Called automatically when an error occurs during program execution.

The format for enabling **DEBUG** is:

DEBUG (PARAMETER) **«ENTER»**

The only parameters are **ON** and **OFF.** If no parameter is specified, **DOSPLUS** will assume **ON.**

Once **DEBUG** is called, you have the following commands:

COMMAND	OPERATION PERFORMED
A	**ASCII**/Graphic display mode
C	Instruction/Call step
DAAAA	Set memory display address to AAAA
GAAAA(BBBB,CCCC)	Go to address AAAA, with breakpoints optionally set at addresses BBBB and CCCC

DEBUG (cont.)

COMMAND	OPERATION PERFORMED
H	Set hexadecimal display mode
I	Single step next instruction
MAAAA	Set memory modification mode starting at address AAAA (optional)
O	Exit to **DOSPLUS** (**DEBUG** still engaged)
RPR AAAA	Alter register pair (PR) to AAAA
S	Set full screen memory display mode
U	Dynamic display update mode
X	Set register examine mode
;	Display next memory page
— (DASH)	Display previous memory page

DEVICE — This function will display on the screen all **I/O** devices and their driver vector address. Simply type **DEVICE** and press **«ENTER»**. The screen will display:

***KI = X'4B9C'**	(keyboard)
***DO = X'0473'**	(video monitor)
***PR = X'4CB4'**	(printer)
***RI = X'301E'**	(**RS232** input)
***RO = X'3021'**	(**RS232** output)

The address is in standard hex notation. There are no parameters for this function. Whenever you use the **FORCE** command to route these, you will notice that the addresses of the driver vectors will change.

DIR — This function will display the directory of the files on the specified disk. To read the directory of the disk in drive 0, simply type **DIR** and press **«ENTER»**. This will give you a listing of all the visible, active files presently on the disk in drive 0. To obtain the directory of a disk other than that in drive 0, you must specify the drive. The format is:

DIR :D **«ENTER»**

NOTE: There is a mandatory blank space between the **R** in **DIR** and the colon **(:)**. If this blank space is omitted, the directory of the disk in drive 0 will be displayed.

If you wish to see the directory of files other than the visible files, you must specify a parameter for the type of file you wish to see. The format is,

DIR :D (PARAMETER) **«ENTER»**

Where the parameter can be:

- **(S)** — System files as well as visible files
- **(I)** — Invisible files as well as visible files (System files will not be displayed)
- **(D)** — All files marked for deletion as well as the visible files.
- **(P)** — Print directory on printer

You can specify more than one parameter.

EXAMPLE **DIR :1 (S,I,D,P) «ENTER»**

This will print on the printer a directory of all the files on the disk in drive 1.

The directory gives you a great deal of information about the disk files automatically. **DOSPLUS** has a unique format. When you call up the directory, you will see a display that looks like this:

```
DIRECTORY DRIVE: 0              DOSPLUS —              02/16/81 — 3.3 DDEN
FILENAME          ATTRIB      LRL   #LOG   #PHY   #GRN   #SEG   EOF
BOOT     SYS      ISB7        256      5      5      1      1     0
DIR      SYS      ISB7        256     10     10      2      1     0
BACKUP   CMD      I*U6        256      9      9      2      1   248
FORMAT   CMD      I*U6        256      9      9      2      1   248
ACCTS    DAT      N*A4         63    600    150     15      1     0
GETTAPE1 BAS      D*X0          1    966      4      1      1   198
SYSO     SYS      ISB7        256     12     12      3      1     0
                  ***     41 GRANS,     51K  ***

DOS PLUS
#
```

NOTE: When the screen is full, the listing will pause until you press either «**ENTER**» to display the next full page or the «**SPACE BAR**» to single-step through the files. The «**BREAK**» key will return you to **DOSPLUS**.

The first information you will see in the directory is the drive number that the disk is in and the date the disk was created. The last item on the first line is the version of the **DOS** used to create the disk. The second line contains the column headings for the directory. The filename column gives you the name of the file and the extension, if any. The next column is the attribute list and is broken up as follows:

The first character is:

N — Normal (visible)
I — Invisible
D — Deleted file

The second character is:

***** — User file
S — System file

The third character is the password attribute and is as follows:

X — No password currently set
A — Access password set
U — Update password set
B — Both access and update passwords set

The last character is the level of protection afforded by the access password. For a listing of these protection levels, see the **ATTRIB** command

The **LRL** column gives the logical record length of the file. The 1 in this column for the program **GETTAPE1/BAS** indicates that this program was saved in the **ASCII** format.

The **LOG** column gives the number of records in the file. For files with a logical record length of **1**, it gives the number of bytes in the file.

The **PHY** column gives the number of sectors used for the file. In the above example, the **966** bytes used four disk sectors.

The **GRN** column gives the number of granules used by the file.

The **SEG** column gives an idea of how the file is distributed on the disk. The **1** in the examples indicates that the files are written on consecutive tracks and sectors.

The final column is the end of file marker for that file.

The final line in the directory is the amount of free space left on the disk in both granules and kilobytes (1024 byte blocks).

DO — This feature begins automatic execution of a file created with the **BUILD** command. The format is:

DO FILESPEC «**ENTER**»

The **DO** command will automatically add the extension **/BLD** to the filespec if no other extension is specified. To **DO** another type of file, you must specify the extension. You can combine the **AUTO** and the **DO** to power up directly into your program. The format for this is:

AUTO DO FILESPEC «**ENTER**»

This can even be done for **BASIC** programs. In fact, **DOSPLUS** has such a powerful **DO** function that you can write a simple **BASIC** program using the **BUILD** command, enter, then execute it directly from **DOS.**

DUMP — This function allows you to down load a section of memory directly to a disk as if it were a program. The format is:

DUMP FILESPEC :D (**START** = X'AAAA',**END** = X'BBBB',**TRA** = X'CCCC')

START — The hex address where the program starts.

END — The hex address where the program ends.

TRA — The hex address where program execution begins. If no transfer address is specified, **DOSPLUS** assumes that the **START** and **TRA** addresses are the same.

If no other extension was specified when the filespec was entered **DOSPLUS** will automatically add the extension **/CIM.**

To execute the program once it has been dumped to the disk, simply enter the filespec from **DOS** using the **LOAD** command, or you can load in **DEBUG,** then the program and execute it from there. If you wish, you can use decimal values in place of the hex in the input line. Just make sure you always use hex notation when needed.

FORCE — This function allows the user to route **I/O** between various devices. Its primary use is to route output that would normally go to the printer to the screen, or the screen to the printer. The format is:

FORCE *DEVICE **TO** *DEVICE
or
FORCE *DEVICE *DEVICE

The parameters are:

DO — Video monitor
KI — Keyboard
PR — Printer

EXAMPLE **FORCE *PR TO *DO «ENTER»**

FORMS — This function controls the print drivers and will allow you to direct output to either the serial or the parallel printer ports as well as tell the printer the size of the paper you are using. It will allow you to use narrow paper in a wide printer without the printer trying to print past the right edge of the paper. The format is:

FORMS (**P** = AA,**L** = BB,**W** = CC,**S**,**LF**,**T**) «**ENTER**»

The parameters for the **FORMS** command are:

P	=	Maximum number of lines per page. Standard 11 inch paper has 66 lines per page.
L	=	Number of print lines on a page
W	=	Number of characters on a line
S	=	Directs all output to serial printer. (It is not necessary to set **S** equal to anything.)
LF	=	Auto linefeed on carriage return. Will generate a linefeed on a carriage return for printers that do not have this feature.
T	=	Performs a top of form on the printer.
NUL	=	Generates a linefeed on NUL/CARRIAGE returns.

Default Values:

P	=	**66** (66 lines per page)
L	=	**60** (print lines to be used per page)
W	=	**132** (characters per line)
S	=	**OFF** (output to parallel printer)
LF	=	**OFF** (NO auto linefeed generated)
T	=	**OFF** (NO top of form)
NUL	=	**OFF** (NO linefeed on NUL/CARRIAGE returns)

NOTE: DOSPLUS will power up from a cold boot with the **FORMS** command set to these default values. To disable the automatic print settings type **FORMS (L=66,P=66).** This is useful when a program has its own driver, such as word processing software. Typing **FORMS** with no parameters will display current settings.

EXAMPLE
```
10 CLS
20 LPRINT"TEXT"
30 CMD"FORMS (T)"
40 LPRINT"NEW HEADER FOR NEXT PAGE"
```

This is a small sample of how to get TOP OF FORM inside your **BASIC** program.

FREE — This function will display a free space map of a disk. The format is:

FREE :D **«ENTER»**

The screen will clear and display the following:

FREE SPACE MAP — DRIVE D

```
00-06   X . ! X X ! X X !  . . ! X X ! X X !  X X
07-13   X X ! X X ! X X ! X X !  . . ! X . !  . .
14-20   . . ! . . ! X X ! D D ! X X ! X X !  X X
21-27   X X ! X . ! X . ! X X !  . . ! L L !  . .
28-34   X X !  . . !  . . !  . . !  . . !  . . !  . .
```

An **X** indicates that a track has been used, a period (.) indicates that it is free, a **D** indicates that the track has been allocated for the directory, and an **L** indicates that the track has been locked out due to a flaw when the disk was formatted.

NOTE: DOSPLUS will detect and display all formatted tracks. It will handle 35 to 80 track disks, depending upon your hardware. If the media is double density, three granules will be displayed for every track. The example is single density (two granules per track).

KILL — This feature will allow you to delete a file from the disk directory and re-allocate the space. The format is:

KILL FILESPEC :D **«ENTER»**

If the file is password protected, you must use the password to **KILL** the file. You will note that you only delete the file from the directory, so if you **KILL** a file accidentally, you can recover the file using the **RESTORE** command. If you have written to the disk since you killed the file, the **RESTORE** command may not work.

LIB — This function will list on the screen all of the library commands. The format is:

LIB «ENTER»

LIST — This function will list the contents of a file on the screen or to a printer. The file is displayed in **ASCII** code with the unprintable characters replaced by a period (.). To avoid this, you must use the **CTL** parameter:

LIST FILESPEC :D **(CTL) «ENTER»**

This will output the control codes in the file. The default is to no control codes.

Or for a list to the printer:

LIST FILESPEC :D **(PRINT) «ENTER»**

NOTE: If the file is password protected, you must use the password.

EXAMPLE **LIST ACCNTS.PASSWORD/DAT:1 (PRINT) «ENTER»**

LOAD — This feature loads a machine-language program from disk into memory. Its primary use is to **LOAD** a file created by the **DUMP** command. The program is run using the **DEBUG** command. The format is:

LOAD FILESPEC :D **«ENTER»**

If the file is password protected, you will have to enter the password before you can load the file. After the file has loaded, you will be returned to **DOSPLUS**.

PAUSE — This function stops execution of a program awaiting operator input. Its primary use is to stop the execution of a **DO** file and wait for the operator to press **«ENTER»**. This is useful when the operator must change disks. However, by using the **CMD"** **"** function of **BASIC**, you can use this function in **BASIC** for the same purpose. The format is.

PAUSE REMARK **«ENTER»**

The remark is the **ASCII** character you want displayed on the screen to tell you the computer is waiting for a response from the operator.

EXAMPLE **PAUSE INSERT PAYROLL DATA DISK. «ENTER»**

PROT — This feature will allow you to alter the disk's protection status using the disk master password. With this, you can lock/unlock all the files on a disk, change the master password (provided of course, you know the old master password), or rename the disk itself. The master password was set when the disk was created, using the **FORMAT** or **BACKUP** commands.

NOTE: The master password for the **DOSPLUS** disk is **PASSWORD.**

The format is:

PROT :D (OPTION) **«ENTER»**

The options are:

RENAME — This function will allow you to assign a new name or extension to a disk file. Only the name and extension of the file are affected by **RENAME**. File content and protection status remain unchanged. If the file is password protected, **RENAME** will require the password. The format is:

RENAME FILESPEC **TO** FILESPEC **«ENTER»**

or

RENAME FILESPEC FILESPEC **«ENTER»**

RENAME cannot be used to change the password. If the new filespec already exists, **DOSPLUS** will print an error message on the screen and return you to the **DOSPLUS** prompt.

RS232 — This feature allows you to display and alter the switch settings on the Serial Interface Board under software control. The format is:

RS232 (PARAMETER1,PARAMETER2,PARAMETER3, ETC)

Where the parameters are:

BAUD	— Can be 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200
WORDS	— Can be 5, 6, 7, or 8
STOPS	— Can be 1 or 2
PE	— Parity enable
PI	— Parity inhibit
EVEN	— Even parity
ODD	— Odd parity
BREAK	— Not ready to receive data
RTS	— Request to send
DTR	— Data terminal ready
NOWAIT	— Tells **RS232** drivers not to wait for a byte

EXAMPLE **RS232 (BAUD = 1200,WORDS = 7,EVEN,STOPS = 2) «ENTER»**

This command is vital when using a serial printer. However, if you do not understand any of these parameters, consult the instruction manual that was included with your serial interface.

If you type **RS232** and do not specify any parameters, the screen will display the current settings of the serial interface, if one is installed.

TIME — This feature allows you to display and set the time. The time is set to 00:00:00 when the system is powered up or on a cold boot. This is how you set the time for the **CLOCK** command. The format is:

TIME «ENTER» (to display the time)

or

TIME HH:MM:SS **«ENTER»** (to set the time)

NOTE: All **DOSPLUS** commands and utilities will re-enter the system without your having to re-set the **DATE** and **TIME.**

VERIFY — This function tells **DOSPLUS** to do an automatic read after each write to the disk. The **VERIFY** function is recommended when you are saving programs of great importance. The format is:

VERIFY (PARAMETER) **«ENTER»**

The parameters for the **VERIFY** command are **ON** and **OFF** only. If no parameter is specified, **DOSPLUS** will assume **ON**. On system power up, the **VERIFY** command is **OFF** unless you turn it **ON.**

UTILITIES

The following utilities are included with **DOSPLUS**. They are executed from the **DOSPLUS** command mode.

BACKUP — Duplicate a disk
CLRFILE — Zeros file without re-allocating space
COPY1 — Single drive copy
CRUNCH — **BASIC** program compression utility
DISKDUMP — Display/Modify a disk sector
DISKZAP — Disk editor
FORMAT — Formats a disk and writes system data
PURGE — Deletes all unwanted files from a disk
RESTORE — Recovers inadvertently deleted files
SPOOL — Printer spooler
TRANSFER — Copies all visible files to another disk
MAP — Maps out diskette, showing file locations

BACKUP — This program will make an exact duplicate of a disk. It will work with either a single or multiple drive system. The format is:

BACKUP «ENTER»

DOSPLUS will then prompt for additional input from the keyboard:

SOURCE DRIVE NUMBER?
DESTINATION DRIVE NUMBER?
BACK DATE (MM/DD/YY)?

DOSPLUS will also allow the following format:

BACKUP :D :D **«ENTER»**

DOSPLUS will then skip the first two questions. If the date has been set using the **DATE** command, it will also skip the date question.

If you specify drive 0 for both the source and destination drive, **DOSPLUS** will automatically do a single drive backup. You will be prompted to insert the source and destination disks, as required.

NOTE: If the destination disk is already formatted to a greater number of tracks than the source disk, **BACKUP** will backup the source disk but leave you the extra tracks, which means you can backup a 35 track to a 40 track disk without it re-formatting and losing the last five tracks.

CLRFILE — This utility will completely zero out a file without deleting the file from the directory. This will, in effect, preallocate space on the disk for a new file with the same filespec as the file that was cleared. The format is:

CLRFILE FILESPEC :D **«ENTER»**

If you don't specify a filespec, the computer will ask which file you want cleared. If the program was password protected, you will be required to use the password to clear the file.

WARNING: When you use **CLRFILE**, the file is completely zeroed. This file is gone; nothing can restore it. BE CAREFUL!

COPY1 — This is a single drive copy utility. It is to be used by single drive users to copy a file from one disk to another. The format is:

COPY1 FILESPEC «ENTER»

If the program is password protected, you will be required to use the password before you can access the file. **DOSPLUS** will prompt you when to insert the source and destination disks into the drive. You can use **COPY1** to copy a file from a non-system disk to another non-system disk by following the on-screen prompts. To use a drive other than 0, specify the drive number after the filespec.

CRUNCH — This is a utility that will remove unnecessary blank spaces and remarks from a **BASIC** program. This function is executed from the **DOSPLUS** command mode, *not* from **BASIC.** The formal is:

CRUNCH FILESPEC :D **TO** FILESPEC :D «**ENTER**»

or

CRUNCH FILESPEC :D FILESPEC :D «**ENTER**»

The program will read the **BASIC** program file and write it back to the disk. The source and destination filespecs *must* be the same if the file is to be written back on top of the source. Also, unlike other compression programs, **CRUNCH** will not remove blank spaces from **DATA** statements or those enclosed within quotation marks.

EXAMPLE **CRUNCH TEST1/BAS:1 TO TEST1/COM:1 «ENTER»**

This will compress the **BASIC** file **TEST1/BAS** on drive 1 and create a new file named **TEST1/COM** on drive 1. Both files will now exist on the disk in drive 1.

NOTE: In some programs there are **REMARK** lines that are referenced by **GOTO** or **GOSUB** statements. When a program has been **CRUNCHED** these lines will be deleted, causing the program to crash. To avoid this, type **(LINE)** after the **CRUNCH** command line. **CRUNCH** will then delete the **REMARK** but *not* the line number of the **REMARK** statement.

EXAMPLE **CRUNCH TEST1/BAS:1 TO TEST1/BAS:0 (LINE) «ENTER»**

DISKDUMP — This is a machine-language disk sector display/modify utility. The format is:

DISKDUMP FILESPEC :D «**ENTER**»

If you didn't enter a filespec when you executed **DISKDUMP, DOSPLUS** will ask:

FILE :

Enter the file name and include all extensions and passwords, if any. Once you have displayed the first sector of a file, you have the following options:

; — Advance one sector
— — Go back one sector
\+ — Advanced to end of file
= — Go to beginning of file
P — Print current sector on printer
M — Enter modify mode

From the modify mode you can:

* Move the cursor up, down, right, or left using the arrows keys on the keyboard.
* Enter new data in **HEX** format
* Exit without making modifications «**BREAK**»
* Exit saving modifications «**ENTER**»

* Zero from cursor position on (press Z key)

To exit **DISKDUMP**, press the **«BREAK»** key twice. This will return you to **DOSPLUS**.

DISKZAP

INTRODUCTION

DISKZAP is a powerful disk sector editor. You should use extreme caution while using **DISKZAP** because it can destroy a disk in a matter of seconds it not used correctly. Micro-Systems Software, Inc. assumes no responsibility tor damages that may occur due to the use or misuse ol this utility.

THINGS TO REMEMBER:

1. **«BREAK»** is the universal abort in **DISKZAP**. It will hall whalever operation is taking place at that moment and return you to the **DISKZAP MENU.** If you are in Ihe modify mode, you will exit that mode and return lo Ihe **MENU** without writing the changes to the disk.

2. **DISKZAP** is written in machine-language. This means that it will access a disk fasler than one written in **BASIC**. But it also means that mistakes happen faster, too. Remember, **DISKZAP** doesn't destroy disks, people do. So use caution when making entries from the keyboard.

3. **DISKZAP** will access any sector of a disk regardless of protection status. It will disregard passwords and protection levels. This is both good and bad. There are tew programs that **DISKZAP** cannot look at, modify, or copy. But it is also easy to change a system tile by accident.

4. There are two types ot data address markers. They are:
   0. Data mark (type 0)
   1. Deleted data mark (type 1)

OPERATION

The formal for entry into **DISKZAP** is:

DISKZAP «ENTER»

You will then see the question **"HOW MANY TRACKS?"**. This is the number ot tracks that you specitied when you formatted the disk. If you press **«ENTER»**, **DOSPLUS** will assume 35 tracks. To reset Ihe track count, press **«BREAK»** and you will be returned to Ihe **"HOW MANY TRACKS?"** prompt. After you have made your track count entry, the screen will clear and display:

```
MODEL III DISKZAP UTILITY — VER 3.0
COPYRIGHT ©1981, MICRO-SYSTEMS SOFTWARE

#    MODE
     ZERO
     COPY
     PRINT
     VERIFY
     FORMAT
     DISPLAY
```

By pressing the up and down arrows keys, you can move the pointer until it points at the option you desire. Then press **«ENTER»** to engage Ihe option indicated. Pressing **«BREAK»** will return you to the **MENU.**

MODE

DISKZAP powers up in the double density mode. However, when you select this option from the **MENU** it will ask **"SINGLE OR DOUBLE"** and then **"SECTOR OFFSET"**. Pressing **«ENTER»** will return you to the **MENU** leaving the **MODE** unchanged. Do not assume things when dealing with the ditterences between single and double density. If you don't know for sure, leave it alone! The format is:

SINGLE OR DOUBLE? PARAMETER **«ENTER»**

The parameters are:

S — Single density
D — Double density

Then it will ask you **"SECTOR OFFSET?"**. If you are using **DISKZAP** on a **DOSPLUS** disk, this value should be 0; if you are using it on a TRSDOS disk the value should be 1. This number is the starting sector number on each track. After the **MODE** has been set, you will be returned to the **MENU.**

ZERO

This utility will set any part of a disk that you specify to **HEX 00.** When you select this utility from the **MENU,** you will be prompted as follows:

DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)
SECTOR COUNT? (DEFAULT VALUE = 1)
TYPE DATA ADDRESS MARK? (DEFAULT VALUE = NONE)

NOTE: Pressing **«ENTER»** will give the default value for each prompt.

COPY

This utility will allow you to do a sector-for-sector copy of a disk, or **BACKUP** a whole disk. When you select this utility from the **MENU**, you will be prompted as follows:

SOURCE MODE? (DEFAULT VALUE = CURRENT MODE)
DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)

You will then see:

DESTINATION MODE? (DEFAULT VALUE = CURRENT MODE)
DRIVE? (DEFAULT VALUE = 1)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)
SECTOR COUNT? (DEFAULT VALUE = 1)

Once again you can achieve the default value by pressing **«ENTER»** in response to the prompt. The source question establishes where you want to begin copying, the destination question establishes where you want to write and the sector count question ascertains how many sectors you wish to copy. If you wanted to copy an entire 35 track disk, your sector count would be 350.

NOTE: DISKZAP will copy any program as long as the tracks and sectors are numbered exactly as you specified. However, it does not create directory entries. Without a directory entry, your program is useless. Do not expect an entry to appear in the directory just because you copied some files from another disk.

PRINT

PRINT will give you a hard copy of what the display option prints on the screen. When you select this option, you will get the following prompts:

DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)
SECTOR COUNT? (DEFAULT VALUE = 1)

For the default value, press **«ENTER»** when each prompt appears on the screen. After you have answered the prompts, **DISKZAP** will display the first sector on the screen and at the same time send it to the printer. It will then clear the screen and display the next sector on the screen while sending it to the printer. It will continue in this fashion until the requested number of sectors have been printed. The printed copy will look like this:

```
000000: FFA4 6D0A 003A 93FB 2020 2020 2050 6174 ..M..:..  PAT
000010: 6368 2070 726F 6772 616D 2066 6F72 2044 CH PROGRAM FOR D
000020: 6F73 706C 7573 204D 6E64 656C 2049 4949 OSPLUS MODEL III
000030: 00C5 6D14 003A 93FB 2020 2020 2050 726E ..M..:..  PRO
000040: 6772 616D 6D65 7220 3A20 4D52 4C2F 4D53 GRAMMER : MRL/MS
000050: 5300 686E 1E00 3A93 FB20 2020 2020 506C S.HN..:..  PL
000060: 6561 7365 206D 616B 6520 6365 7274 6169 EASE MAKE CERTAI
000070: 6E20 7468 6973 2069 7320 7573 6564 206F N THIS IS USED O
000080: 6E20 7468 6520 7072 6F70 6572 206D 6163 N THE PROPER MAC
000090: 6869 6E65 0A09 0909 2020 2020 2020 2020 HINE....
0000A0: 284D 6F64 656C 2049 202D 204D 6F64 656C (MODEL I — MODEL
0000B0: 2049 4949 292E 2020 416C 736F 206D 616B III). ALSO MAK
0000C0: 6520 6365 7274 6169 6E20 7468 6174 2074 E CERTAIN THAT T
0000D0: 6865 0A20 2020 2020 2020 2076 6572 7369 HE.  VERSI
0000E0: 6F6E 206E 756D 6265 7273 206D 6174 6368 ON NUMBERS MATCH
0000F0: 202E 2E2E 00CC 6E28 003A 93EB 2020 2020 .....N(.:..
```

The first column is the address of the displayed sector. The first two digits are the track, the middle two are the sector, and the last two digits denote the **HEX** address of the first byte on the displayed line. The next eight columns are the contents of the eight bytes, starting at the addressed byte, displayed in **HEX.** The **ASCII** value of these eight bytes are displayed immediately to the right. The unprintable characters are replaced by a period (.).

VERIFY

This option will check any or all parts of a disk to make sure that it is readable. When you select this option, you will get the following prompts on the screen:

DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)
SECTOR COUNT? (DEFAULT VALUE = 1)

To get the default values, press **«ENTER»** in response to the prompts as they appear.

If for any reason **DISKZAP** cannot read a sector, it will halt and display an error message and the track and sector on which the error occurred. If you press **«ENTER»**, the **VERIFY** function will continue. Press **«BREAK»** to return to the **MENU.**

NOTE: **VERIFY** will always stop when it encounters a data address mark. Also, while every other option will try five times to read a sector before it gives up, **VERIFY** will try only once. The mere fact that you are checking a disk indicates that it is suspect and the least error should be reported.

FORMAT

This utility is a high speed formatter. It is very fast. However, it gets its speed by not verifying its format or writing system data to the disk. It is *not* intended to replace the **FORMAT** command of **DOSPLUS**. With this option, you can format a track anywhere on the disk. This is useful for fixing CRC errors on unused tracks. When you select this option, you will get the following prompts on the screen:

DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
TRACK COUNT? (DEFAULT VALUE = 1)

Press **«ENTER»** to get the default value or enter the desired parameters.

NOTE: This formatter is intended to be used within **DISKZAP** only. It does not write a bootstrap or a directory to the disk. Without this system information, the disk cannot be used by the operating system.

DISPLAY

This option is the heart of **DISKZAP**. This is the option that will allow you to display and modify any sector on a disk. The display is identical to that of the **PRINT** option. When you select this option, you will get the following prompts on the screen:

DRIVE? (DEFAULT VALUE = 0)
TRACK? (DEFAULT VALUE = 0)
SECTOR? (DEFAULT VALUE = 0)

As usual, press **«ENTER»** to get the default value when the prompt is displayed. After the first sector is displayed, you have the following options:

; — Display next sector
+ — Display same sector on next track
— — Display prior sector
= — Display same sector on prior track
M — Enter **MODIFY** mode

When you select the **MODIFY** mode, you will get a block cursor over the first nibble of the first byte in the upper left hand corner of the screen. You can move this cursor on the screen using the up, down, left, and right arrows keys. You can zero from the current cursor position by pressing the Z key. When you are in the **MODIFY** mode, the keyboard will accept valid hexadecimal characters only; all other entries will be ignored. The input is nibble oriented and you must completely **MODIFY** the nibble you are on before **DISKZAP** will recognize any other key. This includes **«ENTER»** and **«BREAK»**.

After you have finished your modifications to the displayed sector, you may either press **«ENTER»** to write the modified sector back to the disk or **«BREAK»** to exit, leaving the sector unchanged on the disk. In both cases you will be returned to the **DISPLAY** mode. To return to the **DISKZAP MENU**, press **«BREAK»**.

FORMAT — This utility formats a new disk and sets all sectors to zero. It also initializes the bootstrap and directory system information for the operating system. The format is:

FORMAT «ENTER»
or
FORMAT :D **«ENTER»**

The computer will then prompt:

WHICH DRIVE IS TO BE USED?
DISKETTE NAME?
FORMAT DATE?
MASTER PASSWORD?
NUMBER OF TRACKS (35 – 80)? (DEFAULT VALUE = 40)
SINGLE OR DOUBLE DENSITY? (DEFAULT VALUE = DOUBLE)

If you specified the drive when you entered the **FORMAT** command, this question will be skipped. If the date has been set using the **DATE** command, this question would also be skipped. After you respond to all of the questions, **DOSPLUS** will format the disk and write the system data to the disk. Check the disk for errors, lock-out any defective tracks, and write the directory to the disk.

You may also re-format a disk that has been used. However, any data on this disk will be lost. To prevent mistakes, **DOSPLUS** will first check the disk to be formatted to see if it contains data. If it does, it will prompt:

DISKETTE CONTAINS DATA, USE OR NOT?

If you enter a **Y** or **YES** to this prompt **DOSPLUS** will re-format this disk. **N** or **NO** will return you to the **DOSPLUS** prompt. Any other entry will be ignored and **DOSPLUS** will ask the question again.

NOTE: If you re-format a disk, the data on the disk will be lost. There is no way to recover this data once it has been erased.

PURGE — This utility will call up the directory of a disk and ask if you want to delete the files, one at a time. To delete a file, you answer **Y** for **YES**. Pressing **«ENTER»** will tell **DOSPLUS** to keep the file on the disk. The format is:

PURGE :D (PARAMETER) **«ENTER»**

The parameters are:

S — System files
I — Invisible files

PURGE does not need a password. It will delete any file from a disk regardless of its protection level. If you press **«BREAK»** before the last file has been called up, you will exit **PURGE** and leave the remaining files on the disk unchanged.

WARNING: Deleting a system file will cripple **DOSPLUS**, since all system files are interactive. If you do delete a system file, there is no telling what odd things **DOSPLUS** may do. However, if you delete a file you didn't want to delete, the **RESTORE** command may recover it.

RESTORE — This utility will bring a file back from the dead. However, **RESTORE** will not recover a file that has been over-written by another file. Its primary function is to recover a file that has been deleted by accident. The format is:

RESTORE FILESPEC :D **«ENTER»**

NOTE: **RESTORE** will *not* do a global search of all diskettes looking for the filename you gave it. If the deleted file that is to be restored is *not* on the disk in drive zero, indicate in the filespec which disk contains the file, or it will produce a **''FILE NOT IN DIRECTORY''** error.

SPOOL — The spooler enables you to set up areas in memory and on disk that the computer can send data to at a high rate of speed, thereby enabling the program currently running to proceed to the next step, and the text will be printed as the printer is available.

NOTE: The spooler is interrupt-driven; what this means to you is that it will never print any faster than 80 CPS no matter how fast your printer is. Also printing will be temporarily suspended during disk I/O. You spool two ways.

1: Memory only. You set up an area in memory for the spooler to use without specifying a disk file for text overflow. This is the fastest and most common method used in spooler operation. To do this, type **SPOOL (MEM** = however many K you want it to use). When it fills up memory, it will usurp control of the CPU and function as normal during a print operation until enough memory is freed for it to continue spooling. If this happens, don't be alarmed. The spooler is still in control, and will return the CPU to you when it is done with it.

2. Memory with optional disk overflow. This method is used when you will be potentially spooling enormous amounts of text. To use it type **SPOOL** *FILESPEC* **(DISK** = how many K it is to use, **MEM** = how many K it is to use). If you don't give it a filespec, it will use the file **SPOOL/TXT**. If you give it a name and no extension, it will use the extension **/TXT** (i.e., **SPOOL TEST (DISK = 10,MEM = 24)** will spool 24K into memory and overflow up to 10K to a disk file named **TEST/TXT**). This is not quite as fast, because the spooler must interrupt the CPU when turning on the drives to either send text to the disk, or when pulling text off the disk. If the filename is already in existence, the spooler will open the file and write text to it. It starts fresh every time. Don't give it the wrong name, or it might write your payroll data on top of your favorite game.

DISK can be anywhere from 0 to 400 kilobytes. If not specified, it defaults to zero. **MEM** can be from 1 to 32 kilobytes. If not specified, it defaults to four. The number is assumed to be in kilobytes, so enter them that way (i.e., **MEM = 12** for 12K memory buffer area).

Remember, once the spooler is loaded, it will remain in effect until the system is re-booted. It also will not interfere with the **DO** buffer in high memory, so you can activate it from a **BUILD/DO** file.

* * * **CAUTION** * * *

DO NOT remove the disk containing the spool file while the spooler is on. The spooler will write to the new disk at the sectors it thinks the spool file is occupying, overlaying everything in its path. Be careful!

TRANSFER — This utility will copy files from one disk to another. It will only work for the multiple drive user. To transfer invisible files, you must include the parameter (I) after the filespec. Protected files will be transferred as non-protected files. Also, **TRANSFER** will *not* overlay a protected file on the destination diskette with the same name as the file being transferred. The format is:

TRANSFER :D :D «**ENTER**»

NOTE: The system diskette must remain in drive 0 at all times. Also do *not* use the delimiter word **TO** in this command.

MAP — This utility will map out all the files on a diskette, showing what tracks and sectors each file is located on. The format is:

MAP :D

You can specify a (P) option to send the output to the line printer. This is typed in after the drive number. The format is:

NOTE: **MAP :1 (P)**

This will map drive one to the line printer.

EXAMPLE

```
DOSPLUS 03/25/81
SYS6      SYS    18,12—18,17
PURGE     CMD    21,12—21,17
ACCREC    BAS    05,13—15,16
SYS8      SYS    19,00—19,05
CRUNCH    CMD    19,06—19,11
```

The first line is the diskette name and the date it was last formatted or backed up. The three columns are:

COLUMN ONE — FILE NAME
COLUMN TWO — TYPE OF FILE
COLUMN THREE — BEGINNING TRACK, SECTOR TO ENDING TRACK, SECTOR.

NOTE: This information is in decimal. In order for the track and sector numbers that **MAP** displays to work in **DISKZAP**, you must convert them to hex.

BASIC

INTRODUCTION

This **BASIC** is authored solely by Micro·Systems Software. If your program now runs under TRS-80 Disk **BASIC**, it will run under this **BASIC**. However, you may have to make some minor syntax changes in your program statements if the following are used in your program. We also do *not* (in this version of **BASIC**) support the new **CMD** features that were added for the Model III. Most of these are duplicated in some other fashion, and we will add the missing ones in the next version.

1. **PRINT&0 —** Octal conversion is gone. (No one used it anyway.)
2. **CMD''S'' —** To return to **DOSPLUS** you need only enter **CMD** and press **«ENTER»**. The "S" is no longer required. As a matter of fact, if it is used you will get an error message.
3. **CMD''I'' —** To access **DOS** commands from **BASIC** you need only enter **CMD''DOS COMMAND'' and press «ENTER»**. You don't have to use the **CMD''I''** syntax of Radio Shack **BASIC**. (Which didn't work properly anyway!)

ENTERING BASIC

To enter **BASIC**, simply type **BASIC** and press **«ENTER»**. Unlike TRS-80 Disk **BASIC** this **BASIC** does not clear the screen and ask the **FILES** and **MEMORY SIZE** questions. The sign-on message and **BASIC** prompt will appear on the screen.

```
DOSPLUS — EXTENDED Z80 DISK BASIC — VER 1.4
COPYRIGHT ©1981, BY MICRO-SYSTEMS SOFTWARE, INC.

READY
```

Your **BASIC** is now up and running. However, unlike TRS-80 Disk **BASIC** you do not have any files open. The default value in this **BASIC** is **0 FILES**, not **3 FILES** as in Radio Shack **BASIC**. To change this, you enter with the following syntax:

BASIC — Load **BASIC** with no files or protected memory.
BASIC * — Re-enter **BASIC** in the event you exited to **DOS** with your program intact.
BASIC FILESPEC — Load **BASIC** and load and run the specified **BASIC** program.
BASIC -F:3 — Load **BASIC** and open 3 files.
BASIC FILESPEC—F:3—M:61000 — Load **BASIC**, open 3 files, protect memory above 61000, and load and run the specified **BASIC** program.

NOTE: There is a mandatory blank space between **BASIC** and any parameter. Once **BASIC** is loaded, there is no way to set files or protect memory; you must return to **DOS** and re-enter **BASIC** using the proper syntax.

EXAMPLE **BASIC PAYROLL/BAS**—F:4

This will load **BASIC**, and run **"PAYROLL/BAS"** opening 4 files.

BASIC COMMANDS

All of the commands in Radio Shack TRS-80 Disk **BASIC** are in **DOSPLUS** Extended Z80 Disk **BASIC**. The syntax for their use is identical with the exception of some of the **CMD" "** commands. These were explained in the introduction. However, there are some new features that have been added.

CMD — The **CMD** function has been changed to allow you to execute any **DOS** command from **BASIC** and return to your **BASIC** program with your program and all variables intact. By the use of this function you can change a disk and get a directory of the new disk and continue the operation of your program. This can even be done under program control. The format is:

CMD "DOS COMMAND" **«ENTER»**

DI — This is a new command. It is used to delete a line number in a **BASIC** program and insert that line into the program at another point. The syntax is:

DI XXXX,YYYY **«ENTER»**

Where XXXX is the source line number and YYYY is the desfinafion line number.

EXAMPLE **DI** 100,122 **«ENTER»**

DU — This is a new command. It is used to copy a line number in a **BASIC** program to another point in the program. The line will now exist at both the old and the new point. The syntax is:

DU XXXX,YYYY **«ENTER»**

Where XXXX is the source line number and YYYY is the destination line number.

EXAMPLE **DU** 100,122 **«ENTER»**

EDIT — Several new **EDIT** commands have been added to make it much easier to edit your **BASIC** programs. In addition to the standard TRS-80 edit commands, you now have the following:

; (SEMI COLON) — List first line of program
/ (SLASH MARK) — List last line of program
DOWN ARROW — List next line of program
UP ARROW — List preceding line ot program
L — Abbreviation for **LIST (L10 – 20)**
D — Abbreviation for **DELETE (D10 – 20)**

E	—	Abbreviation for **EDIT (E10)**
A	—	Abbreviation for **AUTO (A10,5)**
R	—	Abbrevation for **RUN (R"GAME1/BAS")**
L"	—	Abbreviation for **LOAD (L"T1/BAS:1")**
S"	—	Abbreviation for **SAVE (S"LOAN/BAS")**
K"	—	Abbreviation for **KILL (K"PAY/DAT")**
.(PERIOD)	—	List current line of program
,(COMMA)	—	Edit current line of program

OPEN "E" — This is a new command and is similar to the **OPEN "O"** command in Radio Shack Disk **BASIC** except that it opens a sequential file and places a pointer at the end of the file. All new data is added to the file from this point. Now you can add data to a sequential file without having to load the file into memory, add the new data, and write the file back to the disk.

OPEN "R" — The **OPEN "R"** command has now been expanded to allow you to specify the number of bytes in a record. You may now span the track and sector boundaries of a disk file. You will never waste another byte of disk storage space. As an example, if your record is 45 bytes long you will be able to put 5 records on one sector. The 31 bytes that remain on this sector will be used by the 6th record with the remaining 14 bytes being the first 14 bytes on the next sector. The new format for the **OPEN "R"** command is:

OPEN "R",1,"FILESPEC",45

In Radio Shack Disk **BASIC**, you had to calculate sub-records for yourself, and when you reached 256, **PUT** the whole sector to the disk. Now you can set up a buffer of any length and when you have your data in the butter, write it to the disk. Now, instead of having to take the index key and calculate on which sector your record is located, you need only specify the record number to get the exact record you want. To get the fifth record, do a **GET 1,5** and you will get the fifth record.

RENUMBER — This is a new feature. It is not really a **BASIC** command. However, it can be used from **BASIC** to renumber a program loaded into memory. **RENUMBER** is a **DOS CMD** file that can only be called using the **CMD "RENUM"** function in Z-80 Extended Disk **BASIC**. This renumbering utility will change the number of all **BASIC** lines and all references to these lines in the program. One operation of the renumbering utility that a programmer will find most useful is the **CMD "RENUM",!** function. This will check the line numbering of a **BASIC** program and if it finds any unlisted line number error it will print an **ERROR MESSAGE.** This function only checks a program, it does not renumber it. To use **RENUMBER,** you must first load the program to be renumbered into memory. From the **BASIC READY** prompt the syntax is:

CMD "RENUM",PARAMETER1,PARAMETER2

Where PARAMETER1 is the starting line number and PARAMETER2 is the renumbering increment. If no parameters are given the default value of 10 will be assumed.

EXAMPLE **CMD "RENUM"**,100,5

This will renumber the program presently in memory starting with the first line numbered 100 and all subsequent lines will be incremented by 5. You can also specify where you want the renumberer to start and stop.

EXAMPLE **CMD "RENUM"**,AAAA,BBBB,CCCC,DDDD

Where "AAAA" is the new beginning line number, "BBBB" is the increment, "CCCC" is the old beginning line number, and "DDDD" is the ending line number.

TAB — This is a modified command. If you are using it with a **PRINT** command to place an entry on the video screen, it will work the same way that it does in Radio Shack Disk **BASIC**. However, if you are using it with a **LPRINT** command to print an entry on your printer, you can now specify a **TAB** larger than 64. For example, to print at the 100th character position on your printer, the new syntax would be **LPRINT TAB(100)**"XXX". This new feature will make it much easier to format your printer output.

TRACE — This is a modified command. This is entered exactly as in Radio Shack Disk **BASIC**. However, you now single step through a **BASIC** program one statement at a time. You press the «**ENTER**» key every time you want to execute the next statement. The syntax is **TRON** to turn on the trace function and **TROFF** to turn it off.

REFERENCER — This will allow you to reference your **BASIC** program for line numbers, variables, or keywords. The syntax is:

CMD"REF",K,L,V

This will reference the program for all three. If you specify a P, it will do the same thing, but it will output it to the line printer.

CMD"M" — This feature will instantly display all the variables currently allocated, and what they are set to. All you have to do is from the **BASIC** command mode, type:

CMD"M"

The computer does the rest. As with all **BASIC** utilities, you can hold the listing with shift **↑**, and abort with «**BREAK**».

,V (Variable Save) — This command will enable you to go between two **BASIC** programs with all variables intact. The syntax is easy. Simply add the **,V** to the **LOAD** or **RUN** statement.

EXAMPLE **R"TESTPGM/BAS.PASSWORD:2",V**

This will run **TESTPGM/BAS** off drive two and save all the variables currently set. This is great for index arrays.

NOTE: Any variable set in the text as a literal or through data statements may not be saved. All numeric variables will be intact as well as all variables input from disk or calculated.

EXAMPLE **10 A$=THIS IS A TEST":GOTO 35**

The **BASIC** sets the variable pointer to point to that spot in the text, and when you load in the new text, it will be pointing to something different. When you load the original program back in again, the string will be OK.

SEARCH and **REPLACE** — This is a great programmer's tool. It will search for and display or replace any string variable or expression. All you have to do is type **CMD"SR"** (for search and replace) followed by a literal **ASCII** string, *or* any character string or other string variable. The format is:

CMD"SR",":STOP**","**"

This will go through the text and replace every **:STOP** with a null (mass editing). After it alters a line, it will list that line showing the change. If you type **CMD"SR","**TEST", it will look through the text and list every line with the word "TEST" in it. This is useful for single variable references. It is also great for inserting linefeeds to make text easier to read.

EXAMPLE **CMD"SR","**:**",CHR$(10)+":"**

This will go through the whole text and insert and a linefeed in front of every colon.

TBASIC — TBASIC is **DOSPLUS'** memory saving tiny **BASIC**. The syntax for **TBASIC** is identical to the regular **BASIC** except that it does not contain extended error messages, and it does not allow **DOS** commands from **BASIC** or any of the new shorthand or extended **BASIC** features. The advantage of **TBASIC** is simple; in a 48K machine, after loading in **DOSPLUS** and **TBASIC**, you will have just over 40K of free memory left for your program. With regular expanded **BASIC** you only have about 37K. The sign on message for **TBASIC** is:

TBASIC — EXTENDED DISK BASIC — VER 1.4
COPYRIGHT ©1980, MICRO-SYSTEMS SOFTWARE INC.

READY
#

TBASIC simplified two-letter error messages in addition to the **LEVEL II** error messages. They are:

AD	— File access denied (protected file)
AO	— File already open
BM	— Bad file mode
BN	— Bad file number
DF	— Disk full
DS	— Direct statement in file
EF	— End of file encountered
FE	— File already exists
FF	— File not found
FL	— Too many files (you did not open enough files)
FO	— File overflow
IE	— Internal error
IO	— Disk I/O error
MM	— Mode mismatch (sequential/random files)
NM	— Bad file name
RN	— Bad record number
UE	— Undefined error
UF	— Undefined user function
WP	— Disk write protected

TBASIC gives you the maximum **BASIC** compatibility that we can offer. **TBASIC-F:3** will run almost *anything*. When you are having compatibility problems, try that solution before calling in the troops. **TBASIC 1.4** has also been expanded to add the **",V"** command.

CONVERT

Due to certain incompatibilities between DOSPLUS for the Model III and Model III TRSDOS, there is a program called **CONVERT** that was shipped with your Model III **DOS**. This program also handles the conversion of Model I single density disks.

If a disk is formatted double density on the Model I using the **DOSPLUS** disk operating system, then no conversion is needed. The two systems are 100% disk compatible.

Model I single density and Model III single density are a slight bit different. The Model III does not recognize the data address marks that the Model I creates.

To get around this, type "**CONVERT** :D" where :D is the drive number that contains the target disk. This option will work single drive. The process should take between 2 and 5 seconds. Now the disk is in Model III format, but it is *still* single density. You haven't increased the capacity, you simply made it readable to the Model III.

EXAMPLE **CONVERT :1**

It you wish to then return the disk to the Model I, it may or may not be readable. If your Model I is equipped with **DOSPLUS** for the Model I, NEWDOS/80 from Apparat, or the more recent versions of LDOS from Logical Systems, the Model I will be able to read the disk, with no problem. However, any other system seems to be unable to read these disks.

After you **WRITE** to a single density disk in the Model I you need to re-convert it for the Model III.

To convert a Model III TRSDOS 1.3 disk, insert the disk to be converted in one drive, and the disk you are converting onto in another.

The system disk *must* stay resident in drive zero during this operation. Two drive users *must* convert :1 to :0. Single drive users do *not* have the option of converting Model III TRSDOS disks. After the diskettes are in place, type "**CONVERT** :D :D" where the first :D is the source drive number and the second :D is the destination drive.

Radio Shack changed the directory organization significantly when they went from Model III TRSDOS 1.2 to 1.3. Due to this, you must include an extra parameter to tell the **CONVERT** program which directory it has.

It you are converting a disk that was made using TRSDOS 1.3, after you type the word **CONVERT**, and *after* you specify the drive numbers, type **(V13).** This will tell the program it has a 1.3 diskette. If it is a 1.2 (or 1.1) diskette, you need not include this. The system will assume 1.2.

EXAMPLE **CONVERT** :1 :0 (TRSDOS 1.1 or 1.2)
or
CONVERT :1 :0 (V13) (TRSDOS 1.3)

To take a file back to Model III TRSDOS (either version), do the following:

1. Format a 35 track, single density disk.
2. Copy or save the program to it.
3. Boot up TRSDOS, and use their **CONVERT.**

This will read the file over onto Model III TRSDOS. Please be certain the disk is 35 track, *and* single density (**FORMAT** gives you the options of specifying number of tracks and density).

PATCH PROGRAM INSTRUCTION SHEET

1. GETTING STARTED —

To enter the program type the following:

BASIC PATCH/BAS-F:1

The screen will clear and the program header will display. Make sure the model number matches the machine you are working on.

2. **OPERATION —**

There will be a menu of programs that you can patch. Beside each choice there will be a number. This is the number that you enter for the **"SELECT?"** prompt. If you enter a zero, the program will end and return you to **DOSPLUS.**

When you select the number of the option that you want, the word **"SELECT?"** will be replaced by **"FILESPEC?"**. This is looking for the filename of the program that you want to patch (i.e., **SCRIPSIT/CMD, VC/CMD, INIT**, etc., etc.). It will *not* copy the file over, nor is this filespec the name of what you would *like* to call it. We patch the program, leaving the filename and program location alone.

If you have forgotten what you have named the program, you may type in **"DIR"** when it asks for **"FILESPEC?". This** is *only* valid under Extended Disk **BASIC**, it is not allowed under **TBASIC.** It will ask **"WHICH DRIVE?"**. You respond with the drive number, and it will respond with the proper directory.

When it returns to the **"SELECT?"** prompt and prints the message **"PATCH COMPLETE."**, you have installed the patch. You may, at this point, exit and run your patched program.

NOTE:

Please make certain you patch the **INIT** file in **PROFILE** Model III and not **PROFILE/CMD.** This is the only file that needs a patch.

Patches to MicroSoft's **BASIC** and ForTran compilers are available on request. Patches are also available for the Electric Pencil, ST80III, and many other programs.

Many times the version of the program you have will run just fine. If this is the case, don't worry about it. Often only certain versions of a particular program are incompatible. *Most* programs *will* run "as-is." The ones that don't usually are incompatible in either the end of file handling, or in calling a directory from program.

Before you assume anything, give us a call. We have people available from 10 o'clock in the morning till 5 o'clock at night to help you. Call us at (305) 983-3390.

Thank You,
Micro-Systems Software Inc.
Technical Support Division

```
0     '                TESTPGM — Variable Lengfh Record Example
                       Created : 05/14/81     MRL/MSS
10    CLEAR 500 :DEFSTR A,Z :DEFINT I,N
20    Z1=CHR1(30) :Z2=CHR$(31) :Z3=CHR$(27) :Z4=CHR$(140)
30    CLS:PRINT"Testpgm Ver 1.0" :PRINT"By Micro-Systems Software
40    '
50    '
60    '
70    PRINT@45,Z1;Main menu"; :PRINT  256,Z2;
80    PRINTTAB(10)"1. Input new records" :PRINTTAB(10)"2. Review old records"
      :PRINTTAB(10)"E. End session" :PRINT@
90    PRINTTAB(10)"Selection?" ; :LINE INPUT A
100   IF A="E" OR A="E" THEN CMD
110   IF A<>"1" AND A<>"2" THEN PRINT Z3;Z1;:GOTO 90
120   ON VAL(A) GOTO 160,550
130   '
140   '                Add new records
150   '
160   OPEN"R",1,"TEST/INX",4
170   OPEN"R",2,AS A1,2 AS A2
180   FIELD 1,2 AS A1,2 AS A2
190   FIELD 2,10 AS A3
200   GET 1,1:I1=CVI(A2)
210   '
220   '                Input data
230   '
240   PRINT@45,Z1;"Add records";:PRINT@256,Z2;
250   '
260   LINE INPUT"Record number?";Z1;:GOTO 260
270   IF LEN(Z)>4 THEN PRINTZ3;Z1;:GOTO 260
280   IF Z="" THEN LSET A1=MKI$(0) :LSET A2=MKI$(I1) :PUT 1,1 :CLOSE :GOTO
      70
290   '
300   '                Search index
310   '
320   LO=I1+1:XQ=VAL(Z) :FF%=0 :GOSUB 940
330   IF FF%=1 THEN PRINT"Number already in file." :PRINT Z3;Z3;Z1;:GOTO 260
340   '
350   '                It not in file, bump index
360   '
370   FOR I=I1+1 TO MZ STEP-1 :GET 1,I :PUT 1,I+1
380   '
390   '                Put index record
400   '
410   I1=I1+1 :LSET A1=MKI$(VAL(Z)) :LSET A2=MKI$(I1) :PUT 1,MQ
420   '
430   '                Input data to be filed
440   '
450   LINE INPUT"Remark (10 char. max.)?";Z
460   IF LEN(Z)>10 THEN PRINTZ3;Z1; :GOTO 450 ELSE LSET A3=Z
470   PUT 2,I1
480   '
490   '                Loop back for next
```

```
500  '
510  PRINT CHR$(28);CHR$(256,Z2); :GOTO 260
520  '
530  '                    Review old records
540  '
550  PRINT CHR$(28);CHR$(45,Z1);"Review records"; PRINT CHR$(28);CHR$(256,Z2);
560  OPEN"R",1,"TEST/INX",4
570  OPEN"R",2,"TEST/DAT",10
580  FIELD 1,2 AS A1,2 AS A2
590  EIELD 2,10 AS A3
600  '
610  '                    Get record number and search index
620  '
630  LINE INPUT"Record number?";Z
640  IF LEN(Z)>4 THEN PRINTZ3;Z1; :GOTO 630
650  IF Z="" THEN CLOSE:GOTO 70
660  '
670  GET 1,1:I1=CVI(A2)
680  LQ=I1+1:XQ=VAL(Z):FF%=0:GOSUB 940
690  IF FF%=0 THEN PRINT"Not in file ":PRINTZ3;Z1; :GOTO 630
700  '
710  '                    If found
‡720 '
730  GET 2,CVI(A2)
740  '
750  '                    Display and edit (if needed)
760  '
770  PRINT"Remark —>";A3
780  LINE INPUT"Change (Y/N)?";Z:IF LEN(Z)>1 THEN PRINTZ3;Z1; :GOTO 780
790  IF Z="Y" OR Z="y" THEN PRINTZ3;Z1; :LINE INPUT"New remark (10 char.
     max.)?";Z ELSE GOTO 880
800  IF LEN(Z)>10 THEN Z="Y" :GOTO 790
810  '
820  '                    Save changes
830  '
840  LSET A3=Z:PUT 2,LOC(2)
850  '
860  '                    Loop back for next
870  '
880  PRINT CHR$(28);CHR$(256,Z2); :GOTO 630
890  '
900  '                    Subroutine to do HI-LO search
910  '  Inputs LQ (list length) and XQ (search variable)
920  '  Returns MZ (insert slot)
930  '
940  MQ=INT(LQ/2)
950  IF MZ=0 THEN GOTO 1010
960  GET 1,MQ+1:IF CVI(A1)=XQ THEN FF%=1:GOTO 1020
970  IF CVI(A1)<XQ THEN GOTO 980 ELSE LQ=MQ:GOTO 940
980  TQ=LQ-MQ
990  TQ=INT(TZ/2)
1000 IF TQ=0 THEN 1010 ELSE MQ=MQ+TZ:GOTO 960
1010 MQ=MQ+1
1020 MQ=MQ+1:RETURN
```

```
10    '                Directory Subroutine
20    '                Created 06/09/81     MRL/MSS
30    '
40    CLEAR 1600:DIM A$(128)
50    '
60    '                Input drive and open DIR
70    '
80    INPUT"Which drive";DR$:DR$=":"+DR$:OPEN"R",1,"DIR/SYS"+DR$
90    '
100   '                Read sector
110   '
120   J=0:FOR I=3TOLOF(1):GET 1,I:FOR II=0T07
130   FIELD 1,(II*32) AS D$,1 AS A$,4 AS D$,8 AS FS$,3 AS FE$
140   '
150   '                Check for SYS and DEL files
160   '
170   IF NOT(CVI(A$+CHR$(0))AND208) =16 THEN 220
180   '
190   '                Build DIR array
200   '
210   A$(J)=FS$+"   "+FE$:J=J+1
220   NEXT II,I:CLOSE
230   '
240   '                DONE — print results
250   '
260   FOR I=0T0J:PRINTA$(I),:NEXT
```

```
10    '                Program to input the date from BASIC
20    '                Valid for Extended Disk BASIC ONLY!!
30    '                Created : 09/22/81     MRL/MSS
40    '
50    '
60    '                Input date
70    '
80    LINE INPUT"Input today's date (MM/DD/YY)?";DT$
90    '
100   '                Interrogate for correct form
110   '
120   IF LEN(DT$)<>8 THEN GOTO 150
130   IF MID$(DT$,3,1)<>"/" OR MID$(DT$,6,1)<>"/" THEN GOTO 150
140   GOTO 230
150   PRINT"*** Bad format! ***":GOTO 80
160   '
170   '                Pass date to DOSPLUS system
180   '
190   '                Please note that the space after the word DATE in
200   '                CMD"DATE" is required by the system and may not
210   '                be deleted for any reason.
220   '
230   CMD"DATE"+DT$
```

DISK ORGANIZATION

Your **DOSPLUS** System Diskette contains the **DOSPLUS OPERATING SYSTEM,** which includes a **LIB**rary of Commands, a **DIR**ectory for all files, and the **SYS**tem tables.

The minimum **SYS**tem that is required on any diskette amounts to one **TRACK** and one **GRAN** of information, of which, one track is taken up by the disk **DIR**ectory and one gran is taken up by what is called the **"BOOTSTRAP"** is to let the operating **SYS**tem distinguish between diskettes containing programs or data and blank diskettes.

The **COMMAND LIB**rary is simply a **LIB**rary of utility commands that the **DOSPLUS "SYSTEM"** diskette is capable of doing. This list of commands is displayed by typing **LIB** and pressing the **«ENTER»** key. Because of the amount of space required by these commands and other features, multi-drive owners could use **"FORMATTED"** diskettes for use in drives 1–3. Such **"DATA"** diskettes only use one track and one gran of system information, therefore leaving more space on your diskettes for more or longer programs.

Each diskette is capable of Single- or Double-Sided operation and can have anywhere from 35 to 80 tracks of information depending on the type of drive you own. Each track contains a certain amount of sectors depending upon whether you are operating single- or double-density. If you are operating single-density then each track contains 10 sectors of 256 bytes each. If you are operating double-density then each track contains 18 sectors of 256 bytes each. If you are a Model I owner you are operating a single-density system unless you have or have had a special double-density board installed. If you are a Model III owner then you are operating a double-density system.

Primarily, data read/write are initially at sector boundaries and have exactly 256 bytes. However, **DOSPLUS** allows *user* specified logical length records (with a default of 256) even if it spans two sectors.

DOSPLUS allows *maximum* use of disk space in that if your program is so long as to run out of continuous space on a disk, **DOSPLUS** will split the program into smaller pieces and put them on the diskette wherever there is room and logically tie the pieces together so there will be no interruption to the user's program. In fact, the user will be totally unaware of this happening.

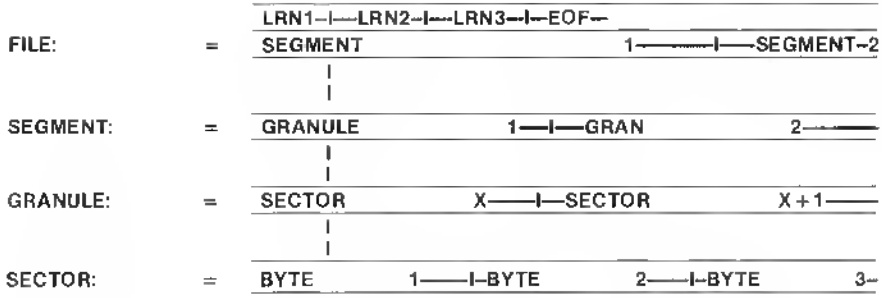
FILE STRUCTURE

A **DOSPLUS** file or **PROGRAM** usually consists of one or more

operating single-density, 5 sectors (5×256=1280 bytes each); for those of you operating double-density density, 6 sectors (6×256=1536 bytes each).

Since a gran is the minimum unit of storage, there are quite a few times when you will not be using, to the fullest extent, all of the room available to you in a file. At these times it is possible to make a few minor additions without adding any more grans to your file. However, you should *never* count on this "extra" storage space.

Each time your program is saved (either initially or when adding to the program), grans are allotted to that program and no matter how long the program is, depending of course that you have enough room on your diskette, the computer will always determine how many grans are needed and put an **EOF** (end of file) marker when the file is closed or saved properly.



LRN: Logical Record Number
The smallest physical unit which can read from or be written to disk.

FILE: Commonly known as a **PROGRAM**
Any number of **LRN**s from the smallest to the largest program length.

SECTOR: A unit containing 256 bytes.

GRANULE: The minimum storage amount of any program containing 1280 bytes (approx. 1.25K) for single-density systems, and 1536 bytes (approx. 1.5K) for double-density systems.

SEGMENT: One contiguous allocation of grans.

SYSTEM ROUTINES FOR ASSEMBLY-LANGUAGE I/O

This information is for those users who have the necessary knowledge of Z-80 machine code. For those who don't, this section is not required reading.

A=XX Register A checks the binary form numeric value of XX and if XX=127 decimal then the value of **A=7FH**. Register is A is used to return error codes to **DOSPLUS** during **I/O.**

B=XX Register B also contains the binary numeric value of XX.

BUFFER This is a buffer, set in **RAM,** that has 256 **USER** designated bytes. The factor that designales whose responsibility it is to manage this area, before and after **I/O,** is the value of the Logical Record Length **(LRL).** If the **LRL = 0** then it is up to the **USER** to control this area. Any other value and the computer is in charge.

DE=>XXXX Register **DE** contains the address of XXXX in machine format.

HL=>XXXX Register **HL** contains the address of XXXX in machine format.

LRL Stands for Logical Record Length. The **USER** can define records of any length between 1–256. If **ZERO** is defined then **DOSPLUS** assumes a **LRL** of 256 bytes.

UREC Stands for **USER** Record. This is the address of the **USER** defined record which is exactly equal to the **LRL.**

A S C I I = American Standard Computers Information Exchange: The same bytes mean the same thing to all American computers. Program must be saved in **ASCII** format by (SAVE"FILENAME",A).

DCB BEFORE OPEN AND AFTER CLOSE

The definition of **DCB** is Device Control Block. The **DCB** is 32 continuous bytes of Random Access Memory **(RAM). DCB** is a left-justified compressed **ASCII** string.

	32
	(FILENAME/EXT.**PASSWORD:D;***)
«NOTE»	Example of a 32 byte **DCB**
	The ; stands for a carriage return
	The * stands for a blank
	Please note the **°EXT.PASSWORD :D** are optional

An explanation of a **DCB** file when it is **OPEN:**

LSB = LEAST SIGNIFICANT BIT
MSB = MOST SIGNIFICANT BIT

EXAMPLE = 6CCO = CO 6C

DCB+ 0	**3**	Reserved
DCB+ 3	**2**	Phy. butter address (LSB/MSB)
DCB+ 5	**1**	Offsef to delimifer af end of Current Rec.
DCB+ 6	**1**	File drive number residence
DCB+ 7	**1**	Reserved
DCB+ 8	**1**	**EOF** last delimiter in lasf Phy. record
DCB+ 9	**1**	**LRL** logical record length
DCB+10	**2**	**NRN** next rec.# open sets **X'0000'** (LSB/MSB)
DCB+12	**2**	**ERN** end rec.# last in file (LSB/MSB)
DCB+14		Reserved

NRN : The nexl record number defines which is fhe next record lo be read/wrillen to/from the file. Each time fhere is a sysfem call, the **NRN** advances by one.

ERN : The ending record number is fhe lasf record number currenfly in fhe file. It is only put info the file when fhe file has been «CLOSE(D)». Therefore the user *must* close his files in order for this fo work correctly.

PHYSICAL AND LOGICAL RECORDS IN DOSPLUS

The definition of a physical record is one sector of disk. A secfor of disk (as you have learned by now) is 256 bytes. If you recall from our previous sfatement in **FILE STRUCTURE,** there are 5 sectors fo a gran in single-density sysfems, and 6 secfors fo a gran in double-densify systems. Wilh Ihis in mind, fhe physical records in a file are numbered: **0** to **N.** The largest record number in a file will be **X** times the number of grans minus one **((X*G)–1).**

Once a file has been opened fhaf a user has already defined from 1 to 255 bytes in lengfh, fhe lengfh cannof be changed unlil fhe file has been closed and reopened with a new **LRL.**

Blocking is puffing more than **LOGICAL** record info a **PHYSICAL** record. This sometimes happens when the **PHYSICAL** length is nof an even mulfiple of fhe **LOGICAL** record length. If the user wishes to do his own blocking all he has fo do is set his **LOGICAL** to zero at the fime he inifially opens his files. Of course, he fhen must figure ouf how to manage his confenfs to fhe **PHYSICAL** record buffer area.

DOSPLUS I/O CALLS

There are al presenl 17 **DOSPLUS I/O** roulines. Here is a list.

FSPEC	**INIT**	**OPEN**	**READ**
WRITE	**VERF**	**POSN**	**BKSP**
PEOF	**FEXT**	**DMULT**	**DIVIDE**
CLOSE	**KILL**	**CKEOF**	**CHKDRV**
FDRIVE			

The **I/O** routines are presenlly described below.

EXIT (JUMP VECTOR 402D)

This is lhe normal exil for **DOSPLUS.**

DEBUG (JUMP VECTOR 440D)

Allows you lo enter **DEBUG.**

ABORT (JUMP VECTOR = **X'4030**)

This is lhe abnormal exil for **DOSPLUS.**

ABORT (JUMP VECTOR = **X'4400**)

Accepts a new command at **DOS** entry.

ERROR (JUMP VECTOR 4409)

Returns an error code message.

A = DOSPLUS

ERROR CODE

HIMEM (JUMP VECTOR = **X'4411**) (MOD **1** = **4409**)

Points to highest **MEM** address in machine.

SET (JUMP VECTOR = **X'4413**) (MOD **1** = **X'4410**)

USER inlerrupl chain

A = Slol # 0–9
DE = Poinl lo inlerrupl
I = Exil

RESET (JUMP VECTOR = **X'4416**) (MOD **1** = **4413**)

Reverse of set

DIR (JUMP VECTOR = **4419**)

Displays file calalog.

FSPEC (JUMP VECTOR = **X'441C**)

This routine checks a filespec for proper name and if it passes moves il to the device conlrol block **(DCB)** so lhe file will be opened. The following represent both enlry and exit condilions which musl be mel.

ENTRY: **(HL)** = > FILENAME
(DE) = **DCB**
EXIT: **Z** = GOOD FILENAME
NZ = BAD FILENAME

INIT (JUMP VECTOR = **X'4420**)

INIT provided the entry poinl in lhe directory afler firsl scanning il to see if the filename is already there. If it is it opens the file for use, and it not lhen it creates a new file with lhe name provided by the device conlrol block **(DCB).** A lypical enlry will look like lhis:

ENTRY: HL=>BUFFER
DE=>DCB
B= LRL
CALL 44204 (HEX)
EXIT: Z SET = OK
A=DOSPLUS ERROR CODE

OPEN (JUMP VECTOR = X'4424)

OPEN pulls the name of the file to be opened directly from the Device Control Block (DCB).

ENTRY: HL=>BUFFER
DE=>DCB
B= LRL
CALL 44244 (HEX)
EXIT: Z SET = OK
Z=0 IF FILE DOES NOT EXIST.
A=DOSPLUS ERROR CODE

READ (JUMP VECTOR = X'4436)

READ first determines the number of the Logical Record Length referred to hereafter as LRL from the Device Control Block. If the LRL is equal to zero the one physical record which was defined at the time the file was opened transfers from the disk file to the RAM buffer ignoring the HL registers and advancing the numerical record number (NRN) by one. If the LRL is greater than zero then the READ transfers the physical record of the number located in DCB to the RAM buffer and advances the NRN by one.

ENTRY: HL=> (IF LRL=0,HL IS UNUSED ELSE HL=UREC)
DE=>DCB
CALL 44204 (HEX)
EXIT: A=DOSPLUS ERROR CODE
Z SET = OK

WRITE (JUMP VECTOR = X'4439)

If the LRL=0 which was defined at I/O, WRITE transfers the physical record from the buffer to the disk and adds the number one to the numerical record number (NRN) which is located in the Device Control Block (DCB).

If the LRL<>0 then whatever record was defined by the NRN is then transferred from the buffer to the disk. At which time the NRN would advance by one.

ENTRY: DE=>DCB
HL=>(IF LRL=0, HL=UNUSED ELSE HL=UREC
CALL 4439H (HEX)
EXIT: A=DOSPLUS ERROR CODE
Z SET = OK

VERF (JUMP VECTOR = X'443C)

The VERF command is very important in that normally whenever you save a program to disk or copy a program from one disk to another, the computer puts whatever material you want to the diskette with no questions asked. However, if something was physically or electronically wrong with the diskette, your material was lost, possibly forever. When VERF writes a physical record to disk, the record is then read back into a special user undefined area, where it is checked byte for byte to make sure whatever you wanted to put there is actually there. The EXIT and ENTRY points are as follows:

ENTRY: **LRL=1** **HL=>UREC**
LRL=0 **HL=**UNDEFINED
DE=>DCB
CALL 443CH (HEX)
EXIT: **Z** SET = **OK**
A = **DOSPLUS** ERROR CODE

REWIND (JUMP VECTOR = **X'443F**)

PARAM JUMP VECTOR = **X'4476)**

BKSP (JUMP VECTOR = **X'4445)**

BKSP accesses the record just before the current record. It the record currently being accessed is the first one in the file, **DOSPLUS** returns with the **NZ** tlag (Invalid Position Warning). The position is not changed.

ENTRY: **D E = D C B**
EXIT: **Z**=INVALID POSITION IN FILE
NZ=INVALID POSITION IN FILE (REC−1)−1

CKEOF (JUMP VECTOR = **X'4457)**

Stands lor Check End ot File.

ENTRY: **DE=DCB**
EXIT: **A=DOSPLUS** ERROR CODE
Z SET ON = NO END OF FILE
Z SET OFF = END OF FILE

GET TIME (JUMP VECTOR = **X'446D)**

This command **GET**s the time and checks tor errors:

HL=ERROR MESSAGE FOR TIME

DISPLAY (JUMP VECTOR = **X'4467)**

This command displays a line ot text:

HL=>TEXT
EOF= CARRIAGE RETURN OR CHR$**(3)**

GET DATE (JUMP VECTOR = **X'4770)** (MOD 1 = **X'4473)**

This command **GET**s the date and checks tor errors:

HL=ERROR MESSAGE FOR DATE.

FDRIVE (JUMP VECTOR = **X'4479)**

Positions the file pointer to the first record in the file, which is handy when the same program must be processed more than once.

ENTRY: **DE=DCB**
EXiT: **Z**=GOOD FILE
NZ=BAD FILE

POSN (ENTRY POINT = **X'4442'**)

Allows access lo any desired logical record in an **OPEN**ed tile. This is a direct mode routine and allows access to data in random order. Following **POSN** with a **READ** or **WRITE** will move data to or trom the logical record. Calculation ot the physical record is done automatically. To **READ** the first record in the file, **POSN** to logical record zero. To add data to the end ot a file, **POSN** to the end of file record number plus one, or **ERN+1.**

ENTRY: **BC** = NUMBER OF THE LRL TO BE SET
DE => **DCB**
CALL 44424 (HEX)
EXIT: **A** = **DOSPLUS** ERROR CODE
Z SET = **OK**

PEOF (ENTRY POINT = **X'4448'**)

This **POSN**s the file pointer to the end of the file. The file might then be check for proper closing, etc.

ENTRY: **DE** = **DCB**
EXIT: **Z** = GOOD FILE
NZ = BAD FILE

DMULT (JUMP VECTOR = **X'444E**)

DMULT does a 16 bit by 8 bit multiplication. The result replaces the multiplicant in **HL**, with the overflow (if any) in the multiplier, **A.**

ENTRY: **A** = MULTIPLIER
HL = MULTIPLICANT
EXIT: **A** = OVERFLOW
HL = PRODUCT

DIVIDE (JUMP VECTOR = **X'4451**)

DIVIDEs a 16 bit dividend in **HL** by an 8 bit divisor in **A.** The quotient is stored in **HL**, and the remainder is placed in register **A.**

ENTRY: **A** = DIVISOR
HL = DIVIDEND
EXIT: **A** = REMAINDER
HL = CMD VALID DRIVE #
C = RETURNS WITH DRIVE #

CKDRIVE (JUMP VECTOR **X'447C**)

CKDRIVE checks to see if drive is available, exists, and if drive is write protected.

ENTRY: **C** = DRIVE INFORMATION
EXIT: **Z** SET = DRIVE EXISTS AND DRIVE #
Z RESET = DRIVE NOT THERE
C SET = WRITE PROTECTED